UNIVERZA V LJUBLJANI

FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Kristjan Jamšek

Sistem za nadzor in upravljanje domače pivovarne

DIPLOMSKO DELO

VISOKOŠOLSKI STROKOVNI ŠTUDIJSKI PROGRAM PRVE STOPNJE RAČUNALNIŠTVO IN INFORMATIKA

Ljubljana, 2017

UNIVERZA V LJUBLJANI

FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Kristjan Jamšek

Sistem za nadzor in upravljanje domače pivovarne

DIPLOMSKO DELO

VISOKOŠOLSKI STROKOVNI ŠTUDIJSKI PROGRAM PRVE STOPNJE RAČUNALNIŠTVO IN INFORMATIKA

MENTOR: viš. pred. dr. Alenka Kavčič

Ljubljana, 2017

Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Tematika naloge: Sistem za nadzor in upravljanje domače pivovarne

V okviru diplomske naloge zasnujte in izdelajte sistem za nadzorovanje in upravljanje procesa varjenja piva. Sistem naj vključuje regulator temperature ter mobilno aplikacijo za oddaljeno upravljanje in spremljanje postopka varjenja piva. Mobilna aplikacija, realizirana na platformi Android, naj nudi intuitiven uporabniški vmesnik za enostavno nastavljanje ustreznih parametrov pri procesu varjenja piva ter omogoča preverjanje posameznih vrednosti temperature preko zunanjih senzorjev in njihovo vizualizacijo. Omogoča naj tudi uvoz, hranjenje, urejanje in izvajanje vnaprej pripravljenih receptov. Za izvedbo regulatorja uporabite platformo Arduino ter izdelajte vso potrebno strojno in programsko opremo.

Zahvaljujem se mentorici viš. pred. dr. Alenki Kavčič, mojim staršem ter zaposlenim podjetij Cosylab in Goap za izkazano razumevanje in pomoč pri pripravi diplomskega dela.

Kazalo

Seznam uporabljenih kratic

kratica	angleško	slovensko
LCD	liquid crystal display	zaslon s tekočimi kristali
EEPROM	electronically erasable programmable read-only memory	električno izbrisljivi programirljivi bralni pomnilnik
SRAM	static random access memory	statični pomnilnik
PWM	pulse-width modulation	modulacija s širino impulzov
SPI	serial peripheral bus	zaporedno zunanje vodilo
PID	proportional-integral-derivative	proporcionalno-integrirno-diferencirni
ROM	read-only memory	bralni pomnilnik
HTTP	hypertext transfer protocol	protokol za prenos hiperteksta
IP	internet protocol	internetni protokol
MAC	media access control	nadzor dostopa do medija
DHCP	dynamic host configuration protocol	omrežni protokol za dinamično nastavitev gostitelja
URL	uniform resource locator	enolični krajevnik vira
SSR	solid state relay	polprevodniški rele
IoT	internet of things	internet stvari
XML	extensible markup language	razširljiv označevalni jezik

Povzetek

Naslov: Sistem za nadzor in upravljanje domače pivovarne

V današnjih časih doživljamo pravi razcvet omreženih naprav, ki klasično niso bile povezane v internet. Kot del novega vala interneta stvari smo v tem diplomskem delu razvili celostno rešitev za nadzor procesa varjenja piva. Celostno rešitev OpenTheBrew sestavljajo uporabniški vmesnik preko mobilne aplikacije Android ter regulator PID in strežnik HTTP, ki ju poganja odprtokodna platforma Arduino Uno v povezavi z razširitveno ploščico Arduino Ethernet Shield. Vsi našteti elementi sodelujejo kot celota v povezavi s temperaturnim senzorjem in električnim grelcem, upravljanim preko releja SSR, z namenom omogočanja uporabnikom enostavno namestitev in upravljanje regulacije PID. Za kar največje udobje in brezskrbnost lahko nastavimo še regulacijske recepte, ki uravnavajo temperaturo vode med procesom varjenja glede na potrebe uporabnika.

Ključne besede: diplomsko delo, regulacija PID, Android, PWM, temperaturna regulacija, strežnik HTTP, Arduino, rele SSR, internet stvari, IoT

Abstract

Title: System for controlling and managing a home brewery

In this day and age we are seeing an ever increasing number of devices being connected into the internet, that were traditionally standalone industrial systems s. As part of the new wave of the internet of things we have developed a technology stack of interconnected solutions to control the brewing process. The OpenTheBrew stack consists of an Android user interface, PID regulator and an HTTP server, both running on the open source Arduino Uno platform, that is connected to the internet via the Arduino Ethernet Shield expansion board. All mentioned elements of the stack work in conjunction with a temperature sensor and a SSR relay, with the intention to give the users a simple way to set up and control PID regulation. For maximum user comfort we can set up regulation recipes, that regulate the water temperature during the brewing process, according to the user's needs.

Keywords: BSc Thesis, regulation PID, Android, PWM, temperature regulation, HTTP server, Arduino, SSR relay, internet of things, IoT

Poglavje 1 Uvod

V sodobnih časih smo priča neverjetnemu tehnološkemu napredku. Poleg tega, da skoraj vsak izmed nas že nosi s seboj pametni mobilnik, bolj zmogljiv od najbolj zmogljivih osebnih računalnikov izpred desetletja, smo priča tudi izjemni pocenitvi raznih senzorskih tipal in mikrokrmilnikov.

Po drugi strani pa nam ves tehnološki napredek ne olajša dela, če ga ne znamo pravilno integrirati v naše aktivnosti. Relativno enostavni in dobro poznani procesi, kot je varjenje piva, se nam kažejo kot idealni kandidati za integracijo sodobne tehnologije s starim znanjem.

Visoko tehnološka oprema, prej na voljo le velikim industrijskim obratom za varjenje piva, zdaj postaja dostopna tudi domačim proizvajalcem in raznim mikropivovarnam. Kar izpade kot mukotrpen ročen postopek ogrevanja vode na določene temperaturne intervale v posameznih korakih lahko pretvorimo v, za uporabnika, enostaven klik na ekranu, ki bo sprožil regulacijski režim.

Namen diplomskega dela je izdelati cenovno ugodno in razširljivo odprtokodno rešitev za samodejni nadzor temperaturne regulacije pri varjenju piva. Kot uporabniški vmesnik naj služi kar pametni telefon z operacijskim sistemom Android, saj ima večinski tržni delež ter je vsa dokumentacija o njemu na voljo brezplačno.

Preden začnemo z načrtovanjem takega sistema, moramo seveda spoznati proces varjenja piva. Dobro razumevanje procesa nam bo omogočilo analizo že obstoječih rešitev v drugem poglavju, iskanje pomanjkljivosti le-teh ter možnosti izboljšav v primerjavi z že obstoječimi sistemi.

V drugem poglavju bomo tudi pripravili seznam zahtev za implementacijo in diagram primerov uporabe, ki nam bodo služili kot visokonivojski načrt razvoja posameznih podsistemov.

Izhajajoč iz primerov uporabe bomo v tretjem poglavju lahko prikazali najbolj primerne strojne platforme za izvedbo posameznih podsistemov. V četrtem poglavju pa bomo predstavili izdelano programsko opremo ter v petem poglavju prikazali, kako z našo aplikacijo izpolnjujemo podane zahteve.

Poglavje 2 Pregled procesa in zajem zahtev za implementacijo

2.1 Proces varjenja piva

Pivo velja za najstarejšo in najbolj priljubljeno alkoholno pijačo na svetu. Pripravljamo jo z varjenjem in fermentiranjem sladkorjev, ki s pomočjo encimov amilaze nastanejo iz škroba. Vir škroba so večinoma razne žitarice, najpogosteje se za te namene uporablja ječmen. Poleg raznih žitaric, so glavne sestavine piva še voda, hmelj in kvas.

Sodobni postopek priprave piva lahko razdelimo na več osnovnih operacij [1]:

- priprava sladu,
- drozganje,
- kuhanje,
- fermentiranje,
- zorenje.

2.1.1 Priprava sladu

Namočena, nakaljena, ter potem posušena ali pražena žitna zrna v pivovarstvu imenujemo slad. Najpogosteje uporabljena so ječmenova zrna, poznamo pa tudi druge vrste piva, ki za pripravo slada uporabljajo pšenico, rž ali oves. V deželah, kjer je riž bolj pogost, lokalni pivovarji uporabljajo tudi riž za osnovo sladu.

Bistvo priprave sladu je sprožitev kaljenja zrn. Kalčki pri kalitvi proizvajajo neaktivne encime α-amilaze ter β-amilaze [2].

S pomočjo encimov α-amilaz lahko razgrajujemo tudi nezdrobljena škrobna zrnca, moramo le zagotoviti dovolj dolg postopek priprave sladu. Najdemo jih med rastlinami, živalimi ter mikroorganizmi. Končni produkt delovanja α-amilaz v sladu je nefermentiran slad.

β-amilaze pa lahko najdemo le v rastlinah. Z njihovo pomočjo lahko razgrajujemo samo že zdrobljena zrna sladu. Med postopkom razgradnje iz škroba pridobimo maltozo in visoko fermentiran slad.

2.1.2 Drozganje

Drozganje je postopek mešanja sladu z vročo vodo z namenom doseganja aktivacije encimov amilaz v drozgi. Proces drozganja se začne pri temperaturi drozge 35 °C, ki jo postopoma segrevamo do temperature 76 °C, kjer pride do zaustavitve delovanja encimov amilaz [2]. Postopek drozganja sicer poteka od ene do treh ur. Z določanjem temperature drozga in časa drozganja lahko pivovarnar prilagodi postopek izbranemu sladu.

Po končanem postopku drozganja se slad odcedi in izpere z vročo vodo temperature od 75 °C do 77 °C. Na ta način izperemo iz zrnja še preostali slad ter vplivamo na nivo tavrina v izprani sladici. Z višjo temperaturo večamo vsebnost tavrina. Večkratno izpiranje sicer vpliva tudi na okus piva, zato je postopek prilagojen posameznemu uporabljenemu receptu.

2.1.3 Kuhanje

Sledi postopek kuhanja, kjer do sedaj pridobljeno sladico steriliziramo in iz nje izparimo vse snovi, ki kvarijo aromo in okus piva. V tem koraku sladici dodamo tudi hmelj, ki pivu da značilno grenkost. Kuhanje navadno poteka od ene do dveh ur [2].

2.1.4 Fermentiranje

Prekuhano sladico prelijemo v posebne posode za fermentacijo, kjer se ohladi na temperature med -2 °C do 26 °C, odvisno od uporabljenega recepta. Tu dodajamo kvas, ki preko kvasovk povzroči alkoholno vrenje [2]. Na tej stopnji lahko fermentirani sladici že rečemo mlado pivo.

2.1.5 Zorenje

Mlado pivo potem zorimo kar ustekleničeno ali v posebnih posodah za zorenje, za čas med dvema tednoma do nekaj let. Tu pivo ohladimo blizu 0 °C, da pospešimo strjevanje beljakovin v mladem pivu ter izločanje neželjenih snovi [2].

2.2 Obstoječe rešitve za nadzor procesa izdelave piva

Na trgu že obstaja nekaj regulatorskih rešitev za varjenje piva. Kot najpopularnejši velja izpostaviti BrewPi in BrewBit.

2.2.1 BrewPi

Rešitev BrewPi je sestavljena iz dveh glavnih delov [3]:

- Računalnik Raspberry Pi, ki služi kot spletni vmesnik za nadzor in spremljanje delovanja regulacije,
- Regulacijski del, osnovan na platformi Arduino, ki skrbi za branje podatkov s senzorjev in krmiljenje izhodov sistema.

BrewPi sestavlja tudi dodaten prikazovalnik LCD, kjer se izpisujeta trenutna in željena temperatura.

BrewPi omogoča nastavljanje regulacijskih receptov, ki vsebujejo več posameznih regulacijskih korakov, kjer ima vsak korak nastavljeno želeno temperaturo in trajanje. Ustvarjeni recepti se shranjujejo na Raspberry Pi delu rešitve, kar pomeni, da v primeru izpada delovanja sistema Raspberry Pi, sistem BrewPi lahko izvaja le trenutni korak regulacijskega recepta.

Ne omogoča pa samodejnega nastavljanja pravil regulacije PID ali uvoza receptov v formatu BeerXML.

Raspberry Pi del rešitve BrewPi med drugim skrbi tudi za beleženje vrednosti, saj ima platforma Arduino premalo pomnilnika za hranjenje tako velike količine podatkov.

Potrebno je poudariti, da je uporaba sistema BrewPi prednostno namenjena tehnično bolj spretnim uporabnikom. Zaradi narave uporabljenih sestavnih delov sistema mora uporabnik opraviti še precej dela v ukazni vrstici, preden je sistem pripravljen za uporabo.

Cena paketa BrewPi znaša približno 200€.

2.2.2 BrewBit

V primerjavi z BrewPi je rešitev BrewBit precej bolj enostavna za uporabo. Proizvajalec jo ponuja kot sistem »priključi-in-igraj«, sposoben krmiljenja dveh ogrevalnih ali hladilnih naprav naenkrat [4].

Za razliko od BrewPi ne ponuja lokalno dostopnega spletnega vmesnika, temveč je potrebno vsako enoto registrirati na centralnem strežniku, ki za nas prikazuje podatke na spletu.

Prav tako ne moremo nastavljati temperaturnih profilov in drugih parametrov za regulacijo brez dostopa do interneta in centralnega registracijskega strežnika. Podobno kot pri sistemu BrewPi, sistem BrewBit omogoča nastavljanje regulacijskih receptov, ki so razdeljeni na več korakov. Vsak posamezen korak sestavlja želena temperatura ter trajanje koraka.

Ker je sistem BrewBit odvisen od delovanja centralnega strežnika nam lahko v primeru izpada internetne povezave preneha delovanje celotnega sistema.

Prav tako kot BrewPi sistem, BrewBit ne omogoča samodejnega nastavljanja pravil regulacije PID ali uvoza receptov v formatu BeerXML.

Cena sistema BrewBit znaša okoli 100€.

2.3 Seznam zahtev za implementacijo

Pri pregledu procesa varjenja piva in konkurenčnih izdelkov smo ugotovili katere minimalne funkcionalnosti naj naš sistem podpira:

- Podatek o temperaturi vode.
- Možnost nastavljanja vnaprej sestavljenih receptov z željeno temperaturo in s skupno dolžino najmanj treh ur.
- Možnost regulacije temperature vode z največjo napako ene stopinje celzija od željene temperature s pomočjo regulacije PID.
- Samodejna nastavitev regulacijskih pravil regulatorja PID.
- Uvoz regulacijskih receptov v formatu standarda BeerXML.
- Neposredno krmiljenje relejev za nadzor električnega grelca.
- Beleženje izmerjenih vrednosti.
- Enostavna uporaba.
- Nizka cena.

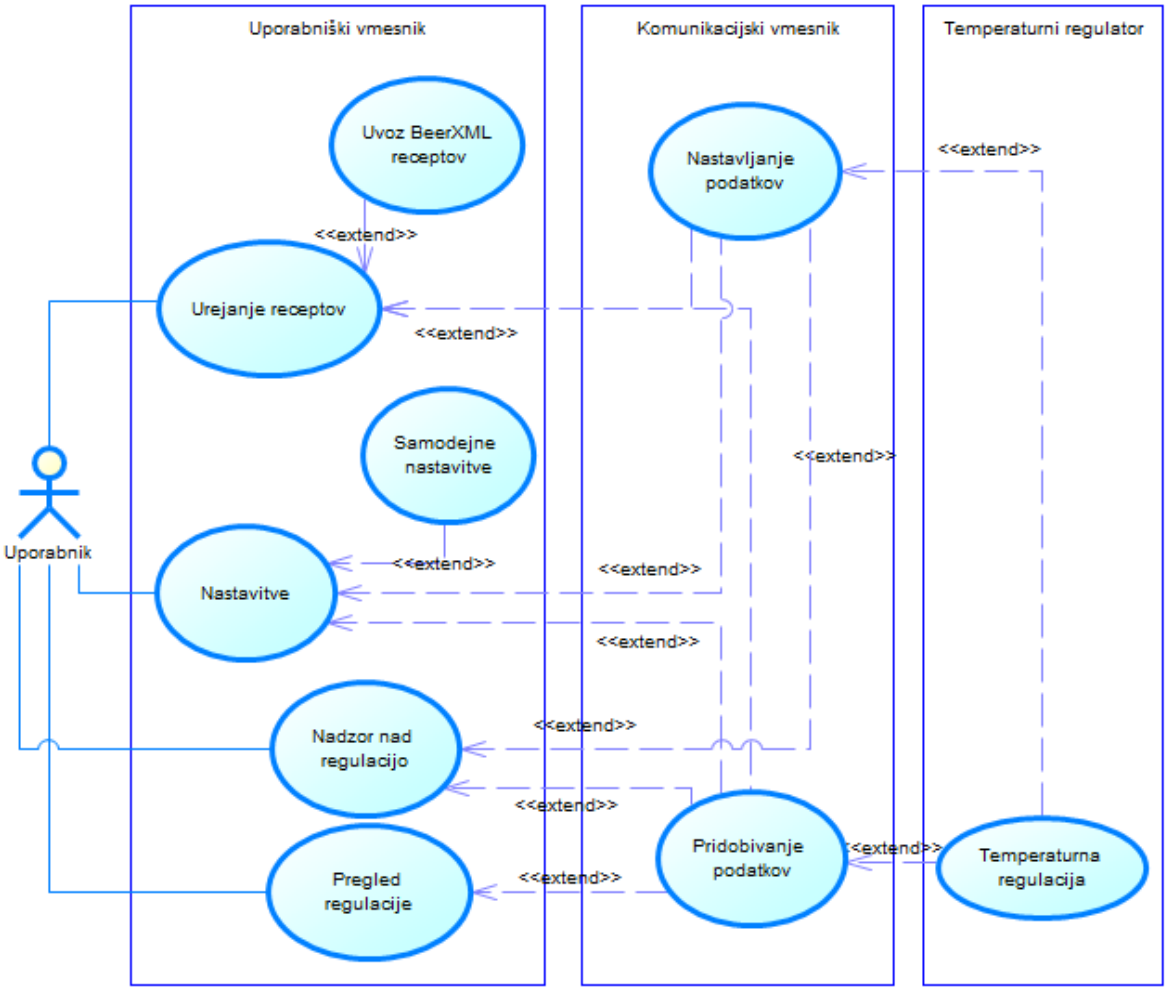
Končni sistem bo tako imel naslednje konkurenčne prednosti pred sistemoma BrewPi in BrewBit:

- Znatno nižja cena.

- Samodejna nastavitev regulacijskih pravil regulatorja PID.
- Uvoz regulacijskih receptov v formatu standarda BeerXML.
- Enostavna namestitev in uporaba.
- Neodvisnost sistema od zunanje internetne povezave, saj bo rešitev delovala le po lokalnem omrežju.

2.4 Diagram primerov uporabe

Izhajajoč iz seznama zahtev smo na sliki Slika 1: Diagram primerov uporabe. sestavili diagram primerov uporabe:



Slika 1: Diagram primerov uporabe.

Iz diagrama uporabe je razvidno, da poleg že definiranega uporabniškega vmesnika v operacijskem sistemu Android potrebujemo še strojno platformo, primerno tako za omrežno komunikacijo kot za temperaturno regulacijo. Kot smo spoznali pri pregledu obstoječih rešitev, bi lahko implementacijo opravili na platformi Raspberry Pi, vendar smo se odločili za platformo Arduino.

Platforma Arduino je v celotni odprtokodna ter relativno enostavna za programiranje v primerjavi s klasičnimi mikroprocesorji. Podpira tudi obilo odprtokodnih knjižnic, ki nam v povezavi z razširitvenimi ploščicami omogočajo enostavno implementacijo strežnika HTTP, ki bo prevzel vlogo komunikacijskega vmesnika. Dodatno lahko na platformi Arduino implementiramo tudi neposredno regulacijo temperature preko povezanega senzorja in grelca.

Poglavje 3 Uporabljena strojna oprema

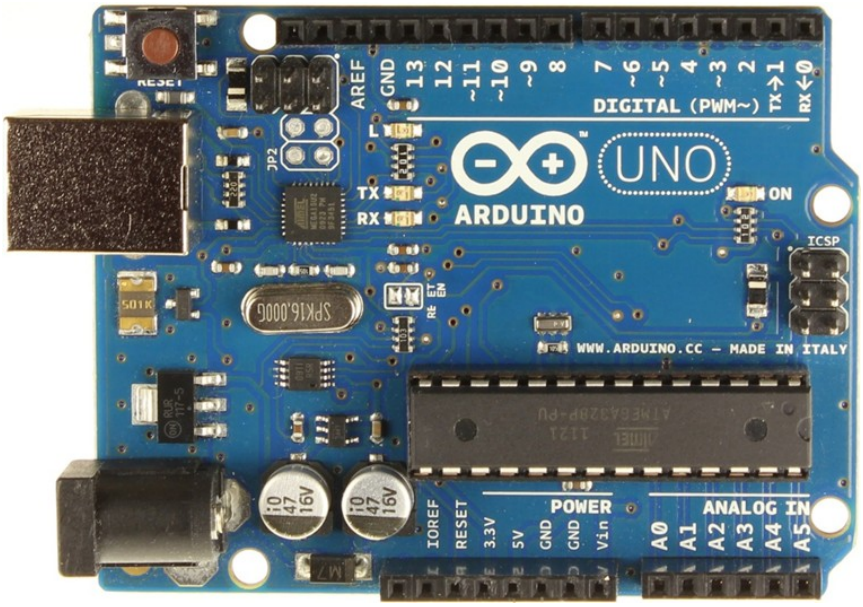
3.1 Arduino Uno

Arduino Uno, prikazan na sliki Slika 2: Mikrokrmilnik Arduino Uno., je odprtokodni mikrokrmilnik posebej zasnovan za enostaven razvoj prototipov. Zaradi nizke cene in enostavne uporabe je idealna izbira za učenje programiranja mikrokrmilnikov. Poganja ga 8-bitni čip Atmel Atmega328P z 32 KB pomnilnika Flash ter z 1 KB EEPROM in 2 KB pomnilnika SRAM [5].

Pomnilnik Flash se večinoma uporablja za shranjevanje prevedene programske kode, pomnilnik SRAM je uporabljen za shranjevanje dinamičnih spremenljivk, medtem ko je pomnilnik EEPROM namenjen za shranjevanje podatkov, za katere želimo, da se ohranijo med prekinitvami napajanja.

Arduino Uno ima na voljo 14 digitalnih vhodno-izhodnih pinov, katerih stanja lahko beremo ali nastavljamo. Vsak pin deluje z napetostjo 5V in maksimalnim tokom 40 mA. Izmed 14 pinov je pin 13 namenjen aktivaciji vgrajene diode LED, 6 pinov (pini 3, 5, 6, 9, 10 in 11, na ploščici označeni s simbolom ~) pa lahko uporabimo za izhode PWM.

Arduino Uno omogoča tudi dodatno uporabo 6 analognih vhodov, ki so na ploščici označeni od A0 do A5. Vsak analogni vhod lahko detektira napetosti med 0 in 5 V z 10-bitno ločljivostjo.

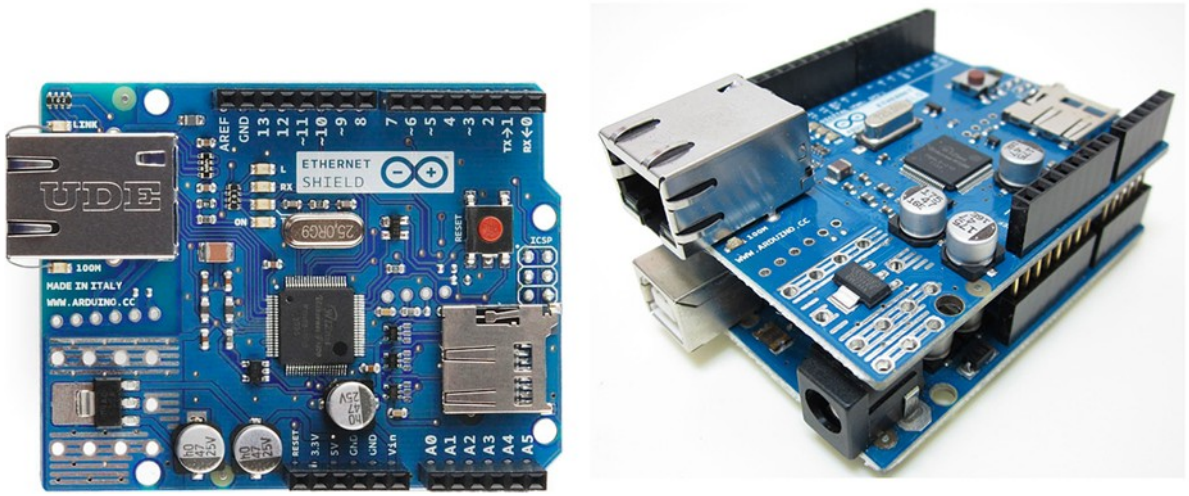


Slika 2: Mikrokrmilnik Arduino Uno.

3.2 Arduino Ethernet Shield

Arduino Ethernet Shield, prikazana na sliki Slika 3: Na levi je prikazana razširitvena plošča Arduino Ethernet Shield, na desni pa sestavljeni Arduino Uno in plošči Ethernet Shield., je razširitvena ploščica za Arduino Uno, ki nam omogoča komunikacijo po omrežju Ethernet. Implementiran je z uporabo Ethernet čipa Wiznet W5100. Na čipu imamo na voljo robustno implementacijo internetnega sklada, ki podpira tako protokol TCP kot protokol UDP. Namestitev Ethernet Shielda je zelo enostavna, moramo le vtakniti Ethernet Shield na vrh ploščice Arduino Uno, tako da so pini na dnu Ethernet Shielda nameščeni v konektorje pinov na ploščici Arduino Uno [6].

Ethernet Shield nam poleg komunikacije preko omrežja Ethernet omogoča tudi uporabo pomnilniških kartic microSD, v primeru če želimo hraniti obsežnejšo količino podatkov. Komunikacija med Ethernet Shieldom in Arduino Uno poteka preko vodila SPI kot tudi komunikacija med microSD ter Arduino Uno. V ta namen moramo uporabiti pina 10 in 4, s stanjem pina 10 specificiramo uporabo komunikacije Ethernet, s stanjem pina 4 pa dostop do kartice microSD.



Slika 3: Na levi je prikazana razširitvena plošča Arduino Ethernet Shield, na desni pa sestavljeni Arduino Uno in plošči Ethernet Shield.

3.3 Temperaturni senzor DS18B20

Digitalni temperaturni senzor DS18B20, proizvajalca Maxim Integrated, je pogosta izbira med temperaturnimi senzorji za platformo Arduino. Z njim lahko komuniciramo preko protokola 1-wire [7].

Vsak temperaturni senzor družine DS18B20 ima unikatni 64-bitni naslov, ki je vnešen v senzorjev pomnilnik ROM med proizvodnjo, preko katerega nam omogoča naslavljanje posameznega senzorja na vodilu. Na ta način lahko z uporabo več temperaturnih senzorjev pokrijemo skoraj poljubno število merilnih točk v posameznem regulacijskem primeru samo z enim mikrokrmilnikom.

Temperaturni senzor DS18B20 omogoča merjenje temperature na razponu od -55 °C do 125 °C z natančnostjo 0,5 °C med -10 °C in 85 °C. Dodatno lahko nastavimo še ločljivost senzorja, in sicer med 9 in 12 biti. Ker ima ločljivost senzorja vpliv na čas vzorčenja, in s tem odzivnost programa, lahko najprimernejšo vrednost najdemo preko tabele Tabela 1 [8]:

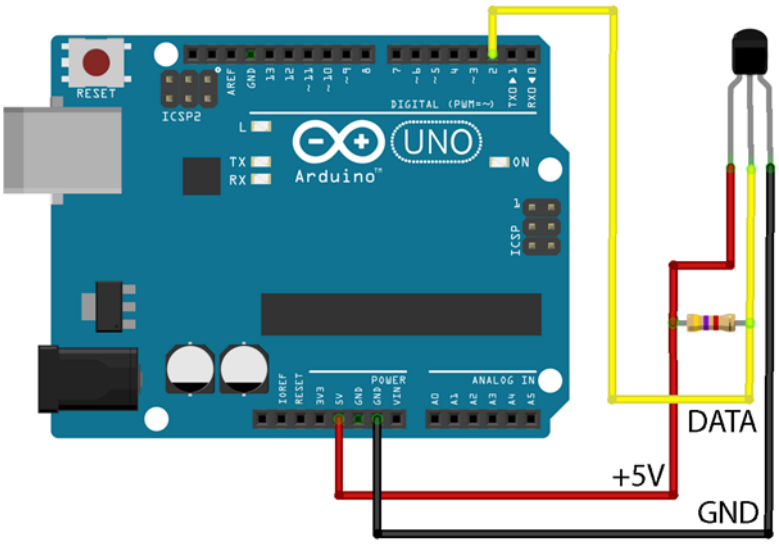
Ločljivost	Inkrement	Čas meritve
9 bitov	0,5 °C	93,75 milisekund
10 bitov	0,25 °C	187,5 milisekund
11 bitov	0,125 °C	375 milisekund

12 bitov	0.0625 °C	750 milisekund

Tabela 1: Odvisnost potrebnega časa za meritve v sorazmerju z ločljivostjo senzorja.

Specifikacije senzorja določajo, da je potrebno, v primeru daljših žic oziroma uporabe več senzorjev na eni žici dodati 4,7 kΩ upor med podatkovno in pozitivno linijo senzorja. Za napajanje lahko uporabimo vir enosmerne napetosti jakosti od 3 V do 5 V.

Končna vezava temperaturnega senzorja DS18B20 na Arduino Uno oziroma Arduino Ethernet Shield je prikazana na sliki Slika 4:



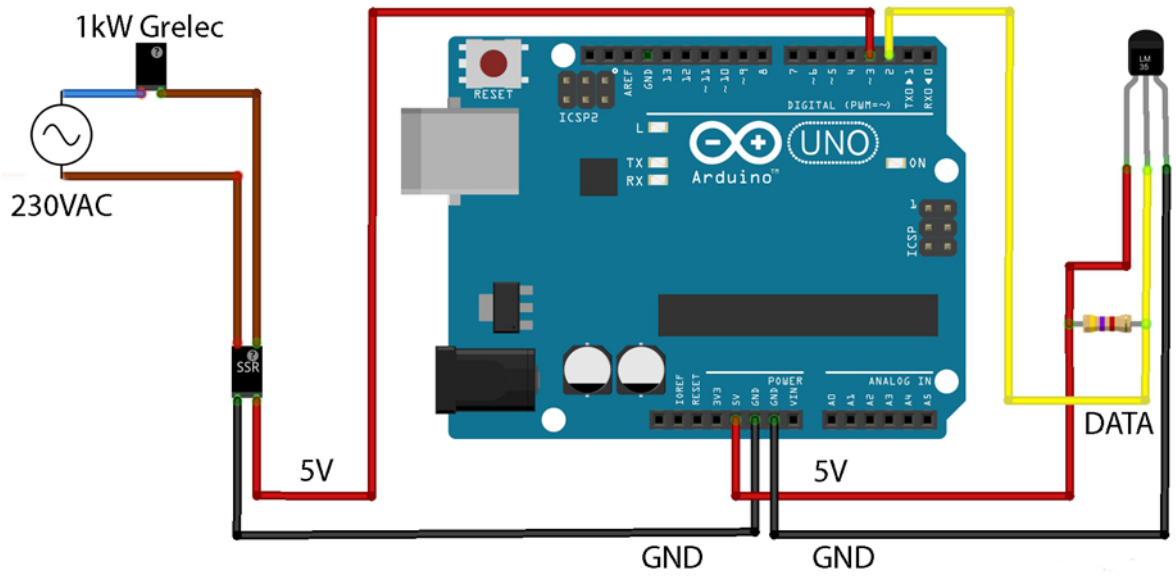
Slika 4: Vezava temperaturnega senzorja DS18B20 na Arduino Uno.

3.4 Fotek SSR 25DA

SSR-25DA, proizvajalca Fotek, je polprevodniški (»Solid State«) rele z možnostjo preklapljanja ločenega tokokroga z izmenično napetostjo od 24 do 380 VAC. Na vhodu releja lahko stanje preklapljamo z neposredno napetostjo jakosti od 3 do 32 VDC.

Za razliko od mehanskih relejev so polprevodniški releji bolj primerni za regulacijske sisteme, kjer lahko pride do hitrih preklopov stanj. Najpogosteje so uporabljeni v raznih regulacijskih sistemih PWM, saj je življenjska doba mehanskega releja pri tej tehniki regulacije občutno manjša. Hkrati s frekvenco preklopov pa narašča tudi segrevanje releja, zato ima rele na zadnjem delu mesto, kjer ga lahko pritrdimo na hladilno rebro.

Gretje vode je bilo doseženo z uporabo 1kW električnega grelca, katerega krmili rele glede na stanja vhodov na releju. Rele, Arduino Uno in električni grelec so povezani kot na sliki Slika 5:



Slika 5: Vezava grelca, releja ter temperaturnega senzorja na Arduino Uno oziroma Ethernet Shield.

Poglavje 4 Razvoj programskega sklada OpenTheBrew

4.1 Uporabljena programska oprema

Pri razvoju aplikacije OpenTheBrew smo uporabili več odrtokodnih programskih knjižnic, ki so nam v veliki meri olajšale implementacijo posameznih delov aplikacije. Na grobo lahko uporabljene programske knjižnice delimo na dva dela:

- Programske knjižnice za platformo Arduino, spisane v programskem jeziku C.
- Programske knjižnice na platformi Android, spisane v programskem jeziku Java.

4.1.1 Programska oprema na platformi Arduino

Vse programske knjižnice, uporabljene na platformi Arduino, so enostavno dosegljive preko orodja Arduino IDE. Z uporabo upravljalnika knjižnic lahko enostavno najdemo in z enim klikom namestimo želeno knjižnico. V naslednjih podpoglavjih bomo opisali namen uporabe posamezne knjižnice ter katere funkcije smo uporabili pri razvoju Arduino dela aplikacije OpenTheBrew.

4.1.1.1 Knjižnica EEPROM

Knjižnica EEPROM nam služi za pisanje in branje podatkov na vgrajeni pomnilnik EEPROM Arduino Uno. Na voljo imamo le en KB pomnilnika EEPROM, ki smo ga uporabili za hranjenje podatkov, katere želimo ohraniti tudi v primeru izpada napajanja.

Funkcija `EEPROM.Get` sprejme dva parametra. Prvi je začetni pomnilniški naslov, s katerega želimo prebrati posamezne podatke. Drugi parameter pa je spremenljivka, ki bo hranila podatek s pomnilniške lokacije. Funkcija sama vrne le referenco na želen podatek, tako da mora programer poskrbeti za skladnost podatkovnega tipa, shranjenega na tej lokaciji, z želeno spremenljivko oziroma podatkovno strukturo.

Funkcija `EEPROM.Put` pa nam služi za vnos podatkov v EEPROM. Ravno tako kot funkcija `EEPROM.Get` sprejema dva parametra. Prvi je začetni pomnilniški naslov, na katerega želimo zapisati podatke, drugi parameter pa je lahko primitivni podatkovni tip oziroma struktura, ki jo želimo zapisati na podani naslov. Funkcija interno uporablja funkcijo `EEPROM.Update`, ki poskrbi za vnos podatka le v primeru, da se podatek razlikuje od prej vnešenega. Ker operacija pisanja v pomnilnik EEPROM potrebuje 3,3 milisekunde za zaključek in ker je pomnilnik EEPROM omejen na 100000 pisanj na posamezno lokacijo,

nam interna uporaba funkcije `EEPROM.Update` omogoči znatno daljšo življenjsko dobo pomnilnika EEPROM.

4.1.1.2 Knjižnici OneWire in DallasTemperature

Knjižnica OneWire je uporabljena za komunikacijo s temperaturnim senzorjem preko protokola 1-Wire. Pri komunikacijskem protokolu 1-Wire lahko preko ene nadrejene naprave dostopamo do ene ali več podrejenih naprav. Če želimo komunicirati z napravo 1-Wire, moramo le podati število pina, na katerega smo povezali verigo naprav, funkciji `OneWire.onewire`, ki nam kreira objekt za komunikacijo.

Referenco na ta objekt lahko potem uporabimo s knjižnico DallasTemperature za enostavno pridobivanje podatkov meritev temperature. In sicer moramo najprej podati refereno objekta OneWire v funkcijo `DallasTemperature.sensorArray`, ki nam preko funkcije `requestTemperature` odkrije vse povezane temperaturne senzorje na tem vodilu. Podatek o temperaturi s posameznega senzorja pa lahko pridobimo s podajanjem zaporednega indeksa senzorja v funkcijo `getTempCByIndex`. Ta nam vrne izmerjeno temperaturo v stopinjah celzija. Ker smo uporabili le en temperaturni senzor, tej funkciji vedno podamo zaporedni indeks 0.

4.1.1.3 Knjižnica Ethernet

Knjižnico Ethernet smo uporabili za vzpostavitev komunikacije HTTP med platformama Arduino in Android. Ta knjižnica nam omogoča uporabo platforme Arduino kot strežnik HTTP, ki odgovarja na zahteve odjemalca, mobilne aplikacije na platformi Android.

Za uporabo te knjižnice imamo dve možnosti, in sicer:

- Nastavitev statičnih naslovov IP in MAC.
- Pridobivanje omrežnih nastavitev preko strežnika DHCP.

Mi smo se odločili za prvo možnost, saj uporaba funckionalnosti DHCP zahteva dodaten pomnilniški prostor za aplikacijsko kodo na platformi Arduino Uno, ki bi imel za posledico prekoračitev razpoložljve količine pomnilnika za aplikacijsko kodo.

Preden lahko zaženemo strežnik, moramo najprej inicializirati polje šestih bajtov naslova MAC ter instancirati objekt `IPAdress` preko funkcije `ip`, kateri podamo željeni naslov IP.

Ko smo definirali naslova MAC in IP, pa lahko s klicem funkcije `server` objekta `EthernetServer` inicializiramo strežnik na posamezni vtičnici, ki jo podamo kot edini argument funkciji server. Končno moramo le še izvesti klic funkcij `EthernetServer.begin` s parametroma za naslova MAC in IP ter `server.begin`. Po izvršitvi teh funkcij je pripravljen strežnik HTTP za odgovarjanje na prejete zahtevke HTTP.

4.1.2 Programska oprema na platformi Android

4.1.2.1 Knjižnica GraphView

GraphView je odprtokodna programska knjižnica za prikaz grafikonov raznih oblik na platformi Android. Omogoča nam prikaz grafikonov v realnem času, povezavo dogodkov z akcijami, omejitev prikaza grafikona samo za določen del podatkov itd.

Pri razvoju naše aplikacije smo se omejili le na manjšo podmnožico funkcionalnosti te knjižnice, saj želimo prikazati le določen del celotnega intervala regulacije. Osnovni objekt za prikaz grafikona se imenuje `GraphView`. Po inicializaciji tega objekta mu lahko podamo enega ali več izmed objektov `LineGraphSeries`, `BarGraphSeries` ali `PointGraphSeries`.

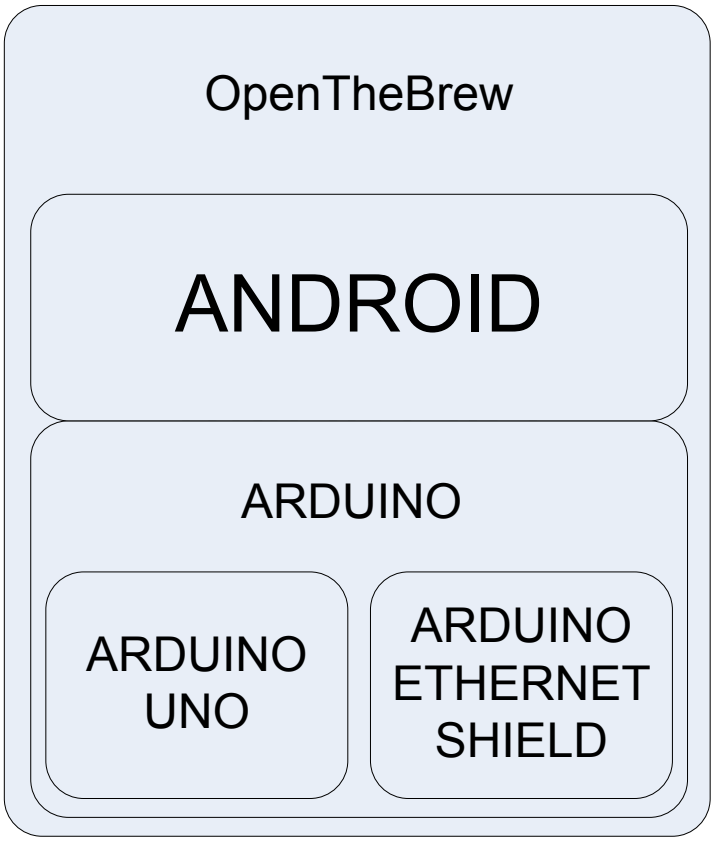
Ti objekti predstavljajo sezname objektov `DataPoint`, ki vsebujejo dva številska podatka, prvega za horizontalno lestvico, drugega pa za vertikalno. Grafikone lahko opremimo s poljubnimi lestvicami, dogodki po pritisku na določen del grafikona ter legendami, kjer opišemo posamezen prikazan podatek.

4.1.2.2 Knjižnica Material File Picker

Material File Picker je odprtokodna programska knjižnica za platformo Android, ki nam v veliki meri olajša delo z implementacijo vmesnika za izbiro datotek. Knjižnico uporabimo kot aktivnost, ki jo zaženemo znotraj neke druge aktivnosti. Ko uporabnik zaključi izbiro datoteke ali prekine brskanje za datoteko, nam ta aktivnost vrne oznako rezultata izvajanja aktivnosti, ki je lahko tekstovni niz z lokacijo izbrane datoteke ali pa ničelna vrednost, ki označuje, da si je uporabnik premislil in zaprl aktivnost brez izbrane datoteke.

4.2 Pregled sestavnih delov sklada OpenTheBrew

Rešitev OpenTheBrew sestavlja sklad povezanih tehnologij, prikazanih na sliki Slika 6.



Slika 6: Diagram programskega sklada OpenTheBrew.

Regulacijski del in komunikacijski vmesnik smo razvili v celoti na platformi Arduino, v programskem jeziku C, z uporabo orodja Arduino IDE. Uporabniški vmesnik pa je bil realiziran na platformi Android v programskem jeziku Java.

4.3 Struktura izvajalne zanke na platformi Arduino

Pri zasnovi izvajalne zanke na platformi Arduino smo uporabili programersko tehniko časovnih rezin. Ta tehnika nam omogoča simuliranje večopravilnosti znotraj sistema, ki tega sicer ne omogoča.

Tehnika časovnih rezin predvideva. da čas izvajanja programa razdelimo na rezine, ki se vedno izvajajo v enakih časovnih intervalih. Pri tem so lahko rezine enakomerne dolžine ali

pa različnih dolžin, paziti moramo le, da ena rezina ne bi vplivala na čas izvajanja drugih rezin. Ravno zaradi tega razloga se je pri uporabi te tehnike najbolje izogniti izvajanju programske logike preko prekinitev.

Na platformi Arduino smo se tu soočili še s težavo, da Arduino privzeto ne podpira delovanja ure v realnem času. Na voljo imamo le klic funkcije `millis`, ki nam poda število milisekund od začetka izvajanja programa. S pomočjo te funkcije lahko enostavno ugotovimo trenutni ter začetni čas izvajanja programa in na podlagi razlike med temi vrednostmi izračunamo, koliko časa je preteklo med klici funkcije.

S pomočjo funkcije `millis` smo v zanki programa na Arduinu definirali tri osnovne časovne rezine:

- Prva časovna rezina:
    - Izvede se vsakih 5 sekund od začetka izvajanja programa.
    - Izvaja računanje vrednosti algoritma regulacije PID.
- Druga časovna rezina:
    - Izvede se vsako sekundo.
    - Izvede primerjavo med analogno vrednostjo, ki jo izračuna prva časovna rezina pri izvajanju algoritma PID, s trenutnim zaporednim izvajanjem druge časovne rezine. Na podlagi te primerjave ustrezno prilagodi izhod funkcije regulirajPWM, ki preko pina 3 upravlja grelec.
- Tretja časovna rezina:
    - V primeru, da trenutno izvajanje zanke ne sovpada s prejšnjimi časovnimi rezinami, rezini 1 in 2 prepustita izvajanje rezini 3. To pomeni, da ta rezina nima fiksnega časa izvajanja kot prejšnji dve.
    - Skrbi za delovanje strežnika HTTP, ki povezuje del rešitve za temperaturno regulacijo s komunikacijskim vmesnikom.

Algoritem izvajalne zanke lahko predstavimo s psevdokodo, prikazano na sliki Slika 7.

```
PID_TICK = 5000
PWM_TICK = 1000
začetni_čas_PID = millis()
začetni_čas_PWM = začetni_čas_PID
while(1){
    trenutni_čas_PWM = millis()
    trenutni_čas_PID = trenutni_čas_PWM

    if(trenutni_čas_PWM - začetni_čas_PWM >= PWM_TICK ){
        reguliraj PWM()
        začetni_čas_PWM = trenutni_čas_PWM
        continue;
    }
    if(trenutni_čas_PID - začetni_čas_PID >= PID_TICK ){
        reguliraj PID()
        začetni_čas_PID = trenutni_čas_PID
        continue;
    }
    obdelajZahtevkeHTTP()
}
```

Slika 7: Psevdokoda algoritma izvajalne zanke.

4.4 Temperaturna regulacija na platformi Arduino

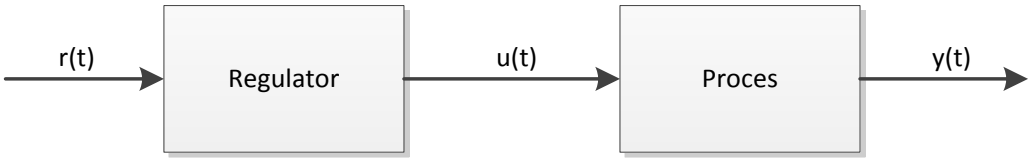
Delovanje vsakega regulacijskega sistema lahko na splošno razdelimo na tri dele: vhodno [9] vrednost v sistem oziroma referenčno vrednost, izhodno spremenljivko, poimenovano tudi procesna spremenljivka, ter delovanje regulacijskega sistema in procesa, ki vplivata na ti dve spremenljivki.

V grobem lahko vse regulacijske sisteme delimo na [9]:

- Odprtozančne regulatorje.
- Zaprtozančne regulatorje.

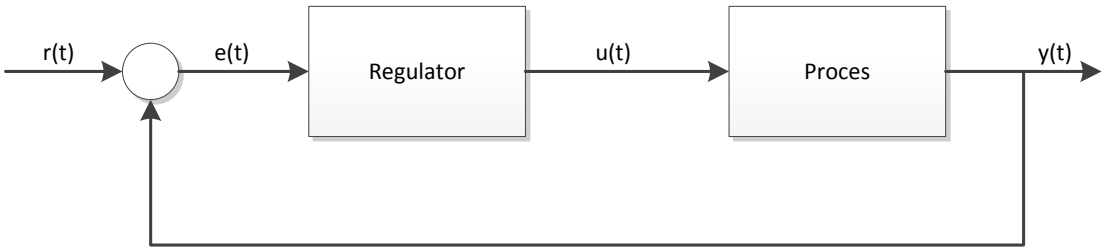
Odprtozančna regulacija ne uporablja povratne informacije pri krmiljenju sistema. To pomeni, da se izhodna procesna spremenljivka y(t) ne primerja z vhodno vrednostjo r(t) pri regulaciji, kot je razvidno iz slike Slika 8. Tak sistem je najbolj primeren za krmiljenje zelo enostavnih procesov, na primer za gretje sanitarne vode v bojlerju. V tem primeru je gretje sanitarne vode zagotovljeno v časovnih intervalih ne glede na dejansko temperaturo vode, saj sistem regulacije nima vednosti o trenutni temperaturi. Za naše potrebe je tak sistem preveč

enostaven, saj ne zadostuje našim zahtevam o največjem odstopanju ene stopinje od želene temperature.



Slika 8: Odprtozančna regulacija.

Pri zaprtozančni regulaciji pa ima regulator vpogled v izhodno procesno spremenljivko y(t). Ta se primerja z referenčno vrednostjo r(t) in razlika teh dveh vrednosti nam predstavlja procesno napako e(t). Regulator na podlagi procesne napake prilagodi svoj izhod v proces u(t) ter tako v veliki meri zmanjša vpliv motenj oziroma ga popolnoma odstrani. Primer delovanja takega sistema lahko vidimo na sliki Slika 9.



Slika 9: Zaprtozančna regulacija.

Tak tip regulacije je idealen za naše potrebe, saj omogoča procesni spremenljivki sledenje referenčni vrednosti sistema. Kot najbolj uporabljen tip zaprtozančne regulacije v industriji se že dolgo uporablja regulacija PID, katero bomo uporabili tudi mi pri izdelavi rešitve.

4.4.1 Regulacija PID

Regulacija PID oziroma proporcionalno-integrirna-diferencirna regulacija je najbolj uporabljen tip odprtozančne regulacije v industriji [10]. Kot smo spoznali pri opisu zaprtozančne regulacije, ta tip regulacije konstantno računa vrednost procesne napake. To določimo kot razliko med referenčno in procesno vrednostjo.

Proporcionalni del je odvisen samo od trenutne procesne napake in ojačitvenega člena. Z velikostjo člena, v razmerju do velikosti procesne napake, lahko povzročimo močan ali šibek

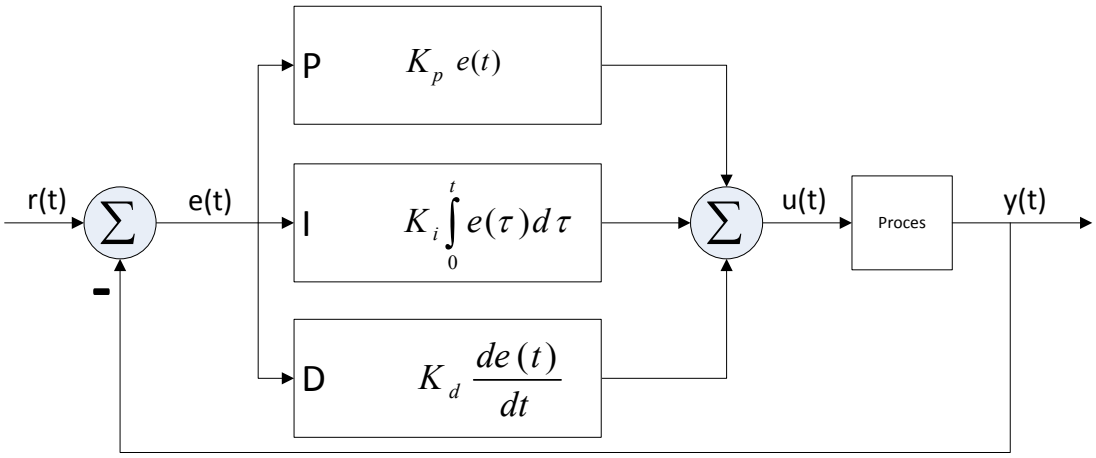
odziv na dano procesno napako. Pomanjkljivost proporcionalne regulacije je v odpravi napake v ustaljenem stanju, saj lahko z veliko vrednostjo proporcionalnega člena povzročamo velika nihanja že pri majhni procesni napaki.

Zato dodamo integrirni del, ki pripomore k zmanjševanju odstopanja v ustaljenem stanju, povzroča pa potencialna nihanja v regulaciji. Dobra lastnost integrirnega regulatorja je, da opravlja regulacijo, tudi ko nimamo trenutne procesne napake, in sicer glede na predhodne napake.

Kot zadnjega dodamo še diferencirni del, ki poskrbi za manjšo oscilacijo pri regulaciji s predvidevanjem spremembe razlike med trenutno in preteklo procesno napako. Na ta način nam lahko pravilno nastavljen člen D pomembno vpliva na dušenje nihanja.

Prilagajanje regulacije sistema glede na vrednost procesne napake in ojačitvenih delov se izvaja kot vsota proporcionalnega (P), integrirnega (I) ter diferencirnega (D) dela.

Regulator PID lahko prikažemo z blokovnim diagramom procesa na sliki Slika 10 [9].



Slika 10: Diagram regulatorja PID.

Formula idealne oblike regulatorja PID je torej opisana s splošno enačbo (1).

$$u(t) = K_p e(t) + K_i \int_0^t e(\tau)d\tau + K_d \frac{de(t)}{dt} \tag{1}$$

Diskretizirana oblika formule regulatorja PID pa je prikazana v formuli (2):

$$u(t) = K_p(e(t) + \frac{1}{T_i}\int_0^t e(\tau)d\tau - T_d\frac{de(t)}{dt}) \quad (2)$$

K_p, K_i in K_d so ojačitve proporcionalnega, interirnega ter diferencirnega dela, T_i nam predstavlja integrirni čas, T_d pa diferencirni čas [11].

Preden lahko zgornjo formulo uporabimo v regulacijskem algoritmu, moramo poenostaviti formuli za integrirni in diferencirni del regulatorja PID. Ker vemo, da se bo regulacijska zanka izvajala v rednih intervalih, lahko integrirni del implementiramo kot kumulativno vsoto trenutne in prejšnjih napak, diferencirni del pa kot razliko med prejšnjo in trenutno napako. Regulacijski algoritem, ki je primeren za programsko implementacijo, lahko vidimo na sliki Slika 11.

```
prejšnja_napaka = 0
integral = 0

while(1){
    trenutna_napaka = željena_vrednost - dejanska_vrednost
    integral = integral + trenutna_napaka
    diferencial = trenutna_napaka - željena_napaka
    izhod = Kp * trenutna_napaka + Ki * integral + Kd * diferencial

    prejšnja_napaka = trenutna_napaka

}
```

Slika 11: Slika psevdokode regulacijskega algorithma PID.

4.4.2 Nastavitev parametrov PID po metodi Ziegler – Nichols

Kot je razvidno iz prejšnjega poglavja, moramo za pravilno regulacijo algoritma PID nastaviti primerne vrednosti parametrov K_p, K_i in K_d. To lahko storimo preko ročnih nastavitev posameznih ojačitev ali pa preko določenih nastavitvenih algoritmov, kot sta metodi Ziegler – Nichols: metoda s pomočjo odziva na stopnico (ta metoda je primerna za odprtozančne procese, zato se nismo osredotočili nanjo) in nihajni preizkus [12].

Pri razvoju naše rešitve smo se omejili na metodo Ziegler – Nichols z nihajnim preizkusom, saj bolj ustreza našemu zaprtozančnemu sistemu.

Metoda z nihajnim preizkusom določa nastavitev ojačitve integrirnega in diferencirnega dela na vrednost 0 ter povečevanje proporcionalne ojačitve, vse dokler ne doseže kritične ojačitve K_{KR}, kjer regulacijski sistem nedušeno zaniha. Iz opazovanja nedušenega nihanja lahko pridobimo vrednost kritične preiode T_{KR}, ki predstavlja interval nedušenega nihanja. Na osnovi teh dveh vrednosti sta Ziegler in Nichols [9] predlagala parametre regulatorja PID, navedene v tabeli Tabela 2:

Vrsta regulatorja	K_p	T_i	T_d
P	0,5 K_{KR}	∞	0
PI	0,45 K_{KR}	0,83 T_{KR}	0
PID	0,6 K_{KR}	0,5 T_{KR}	0,125 T_{KR}

Tabela 2: Tabela nastavitvenih vrednosti po metodi Ziegler – Nichols z nihajnim preizkusom.

Iz splošne enačbe regulatorja PID (1) in diskretizirane oblike te enačbe (2) je razvidna povezava med vrednostmi K_p, K_i, K_d ter T_i in T_d:

$$K_i = \frac{K_p}{T_i} \tag{3}$$

$$K_d = K_p T_d \tag{4}$$

Na podlagi enačb ((3 in (4) lahko ob znani kritični periodi in kritični ojačitvi enostavno samodejno nastavimo ojačitve regulatorja PID. Iz izmerjenih temperatur in period med njimi moramo le ugotoviti, kdaj se v zanki dogodi nedušeno nihanje, ter shraniti izračunane nastavitve v pomnilnik EEPROM.

4.4.3 Nadzor grelca s tehniko modulacije dolžine impulzov

V prejšnjih poglavjih smo spoznali regulacijo PID, kako deluje ter kako lahko primerno nastavimo parametre regulatorja PID za optimalno regulacijo procesa. Preden pa lahko regulacijo PID uporabimo za nadzor stanja električnega grelca, moramo analogno vrednost izhoda algoritma PID pretvoriti v digitalno vrednost, združljivo z binarnim stanjem grelca, saj je grelec lahko le vključen ali izključen.

Za naš sistem se kot najbolj primeren način pretvorbe analogne izhodne vrednosti regulatorja PID v digitalno izkaže uporaba tehnike modulacije dolžine impulzov oziroma s tujko poimenovana modulacija PWM (*Pulse-width modulation*) [13].

Za razumevanje in delovanje modulacije PWM je bistvenega pomena razumevanje pojma služnostnega cikla (*duty cycle*). Služnostni cikel je izražen s procentualno vrednostjo, in sicer predstavlja kvocient med časom, v katerem je digitalni izhod aktiven, ter celotnim časom nekega signala.

V naši rešitvi smo izbrali skupno dolžino cikla 5 sekund, kar ustreza frekvenci izvajanja regulacije PID v prvi časovni rezini glavne zanke. Za namen pravilne izvedbe modulacije PWM smo morali omejiti vrednost izhoda regulatorja PID na maksimalno vrednost 100 in minimalno vrednost -100. V tem primeru nam celotna dolžina signala predstavlja interval dolžine 200.

V drugi časovni rezini vsako sekundo primerjamo vrednost izhoda regulatorja PID in povečujemo števec zaporednega izvajanja rezine dokler ne dosežemo zadnjega zaporednega izvajanja rezine znotraj intervala 5 sekund. V tem primeru števec ponovno nastavimo na 0, saj vemo, da se je skupna dolžina cikla zaključila.

V vsakem zaporednem izvajanju primerjamo vrednost izhoda regulatorja PID z deležem skupnega intervala, ki je odvisen od števca zaporednega izvajanja te rezine. V praksi to pomeni, da ob vrednosti števca 1 primerjamo vrednost izhoda regulatorja PID s številsko vrednostjo -66. V primeru, da je izhod regulatorja PID manjše vrednosti kot -66, nastavimo pin 3 v neaktivno stanje. Če je ta vrednost večja kot -66, nastavimo pin 3 v aktivno stanje. Ob drugem izvajanju rezine primerjamo izhod regulatorja PID z vrednostjo -33 in spet ustrezno nastavimo stanje pina 3. Tako nadaljujemo vse do petega izvajanja zanke, kjer primerjamo izhod regulatorja PID z vrednostjo 66, prilagodimo stanje pina 3 ter nastavimo števec izvajanja rezine spet na 0.

Ker se vrednost izhoda regulatorja PID med izvajanjem druge rezine ne spreminja, lahko na ta način enostavno izračunamo in vzdržujemo pravilen služnostni cikel modulacije PWM.

4.5 Strežnik HTTP z Arduino Ethernet Shieldom

Ko sta prvi dve časovni rezini naše rešitve na platformi Arduino zaključili izvajanje ali pa trenutni čas ne ustreza pogojem prve ali druge rezine, se izvrševanje prepusti komunikacijskemu delu rešitve. Prenos podatkov med uporabniškim vmesnikom in regulacijskim delom rešitve deluje preko strežnika HTTP.

Pri razvoju smo najprej nameravali uporabiti funckionalnost DHCP programske knjižnice Ethernet, vendar smo hitro spoznali, da to pomeni obilo dodatnega zasedenega pomnilnika, namenjenega za aplikacijsko kodo. Zato smo se odločili za inicializacijo strežnika HTTP s statično določenima naslovoma IP in MAC. Preden programsko kodo prevedemo za namestitev na platformo Arduino, moramo nastaviti ustrezna naslova IP in MAC. Naslov IP si moramo zapomniti ter ga vnesti v menu nastavitev v uporabniškem vmesniku.

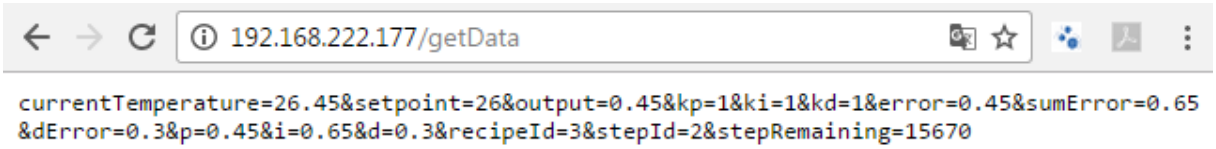
Za postavitev strežnika moramo najprej v inicializacijskem bloku instancirati strežniški objekt. Ob vsakem izvajanju tretje časovne rezine preverimo, če obstaja kaka odjemalska zahteva, kar naredimo s kreiranjem odjemalskega objekta preko klica funkcije available strežniškega objekta. Če noben odjemalec ne čaka na podatke s strežnika, enostavno prepustimo izvajanje spet prvima dvema rezinama.

Ko pa imamo odjemalca povezanega s strežnikom, moramo najprej preveriti, če je pripravljen na komunikacijo, preko funkcije available odjemalskega objekta. Ko je odjemalec pripravljen na pošiljanje, beremo vnosne podatke znak za znakom in, v primeru skupne dolžine manj kot 100 znakov, jih shranjujemo v skupno spremenljivko. Preverba skupne dolžine prejetih znakov je pomembna, da ne prekoračimo omejitve dolžine niza. Končno lahko obdelamo prejete podatke glede na datotečno pot prejetega naslova URL.

Struktura strežnika HTTP je zelo preprosta zaradi pomanjkanja razpoložljivega pomnilnika za aplikacijsko kodo. Namesto več vrst zahtevkov HTTP POST in HTTP GET, kot smo predvideli na začetku razvoja, smo uporabili samo zahtevke HTTP GET tako za pridobivanje kot tudi nastavljanje podatkov.

4.5.1 Primer pridobivanja podatkov o regulaciji

Če želimo pridobiti podatke o regulaciji, moramo poslati zahtevek HTTP GET na naslov IP strežnika HTTP z datotečno potjo /getData. Primer nastavljanja podatkov preko brskalnika lahko vidimo na sliki Slika 12.



Slika 12: Primer zahtevka HTTP GET za pridobivanje podatkov o regulaciji.

Pomen posameznih prejetih podatkov in podatkovnih tipov za vsako spremenljivko lahko vidimo v tabeli Tabela 3: Seznam prejetih spremenljivk o temperaturni regulaciji..

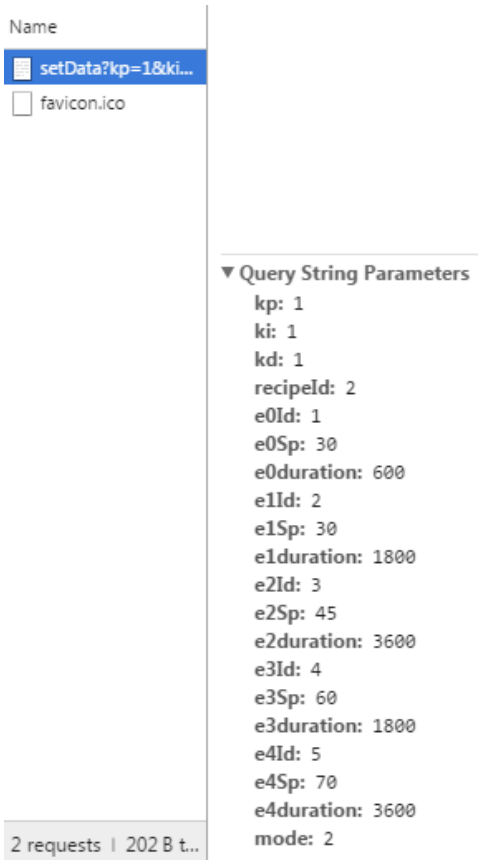
Ime spremenljivke	**Podatkovni tip**	**Opis spremenljivke**
currentTemperature	float	Trenutna izmerjena temperatura povezanega temperaturnega senzorja
setpoint	float	Trenutna želena temperatura
output	float	Vrednost izhoda regulatorja PID
kp	float	Ojačitveni člen proporcionalnega dela regulatorja PID
ki	float	Ojačitveni člen integrirnega dela regulatorja PID
kd	float	Ojačitveni člen difirencirnega dela regulatorja PID
error	float	Vrednost trenutne procesne napake
sumError	float	Vrednost vsote trenutne in preteklih procesnih napak
dError	float	Vrednost razlike med trenutno ter prejšnjo procesno napako
p	float	Vrednost proporcionalnega dela regulatorja PID
i	float	Vrednost integrirnega dela refulatorja PID

d	float	Vrednost diferencirnega dela regulatorja PID
recipeId	int	Vrednost primarnega ključa trenutno aktivnega recepta v podatkovni tabeli receptov
stepId	int	Vrednost primarnega ključa trenutno aktivnega koraka recepta v podatkovni tabeli korakov receptov
stepRemaining	unsigned long	Dolžina preostalega časa izvajanja trenutnega koraka regulacijskega recepta, izražena v milisekundah

Tabela 3: Seznam prejetih spremenljivk o temperaturni regulaciji.

4.5.2 Primer nastavljanja podatkov o regulaciji

Če želimo nastavljati podatke regulatorja, pa moramo poslati zahtevek HTTP GET na naslov IP strežnika HTTP z datotečno potjo /setData?, kateri moramo dodati še imena in vrednosti spremenljivk, ki jih nastavljamo, ločene z znaki &. Primer nastavljanja podatkov preko brskalnika lahko vidimo na sliki Slika 13.



Slika 13: Primer nastavljanja podatkov, zabeležen v razvojnem orodju brskalnika Chrome.

V tabeli Tabela 4 lahko najdemo seznam spremenljivk, katerim lahko priredimo vrednost preko tega zahtevka.

Ime sprejemljivke	Podatkovni tip	Opis spremenljivke
kp	float	Vrednost ojačitvenega člena proporcionalnega dela regulatorja PID
ki	float	Vrednost ojačitvenega člena integrirnega dela regulatorja PID
kd	float	Vrednost ojačitvenega člena proporcionalnega dela regulatorja PID
recipeId	int	Vrednost primarnega ključa trenutno aktivnega recepta v podatkovni tabeli receptov
e0Id ... e4Id	int	Ta nabor vrednosti nam predstavlja vrednosti primarnih ključev korakov izbranega recepta
e0Sp ... e4Sp	int	Ta nabor vrednosti nam predstavlja želene temperature za posamezen korak izbranega recepta
e0duration ... e4duration	int	Ta nabor predstavlja dolžino posameznega koraka izbranega recepta v sekundah
mode	int	Vrednost te spremenljivke določa trenutno stanje regulatorja. Dovoljene vrednosti so: • 0 – Regulator nedejaven • 1 – Regulator ustavljen v trenutnem koraku • 2 – Regulator izvaja izbrani korak • 3 – Regulator izvaja kalibracijo nastavitev regulatorja PID

Tabela 4: Seznam podprtih spremenljivk v regulatorju, katerim lahko priredimo vrednosti.

Kot lahko vidimo v tabeli, smo omejeni le na pet korakov regulacijskega recepta zaradi pomanjkanja prostega pomnilnika za hranjenje večje količine podatkov na platformi Arduino.

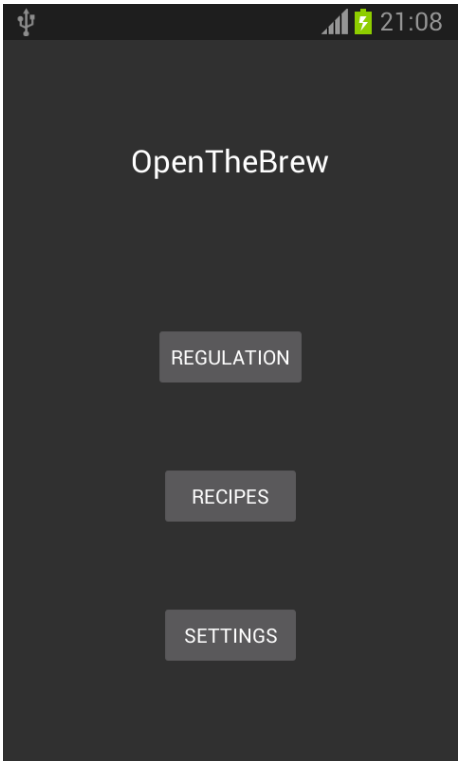
4.6 Uporabniški vmesnik

Kot zadnji manjkajoči del rešitve bomo opisali še uporabniški vmesnik. Odločili smo se za zelo enostaven koncept uporabniške izkušnje, kjer se uporabniški vmesnik deli na tri dele:

- Menu regulacije, kjer spremljamo stanje regulatorja ter nastavljamo želeni recept regulacije.
- Menu nastavitev, kjer lahko ročno, ali samodejno preko metode Ziegler - Nichols, nastavimo parametre regulatorja PID ter vnesemo naslov IP strežnika HTTP.
- Menu receptov, kjer lahko kreiramo, urejamo ter uvažamo recepte za regulacijo.

Vsak izmed ločenih delov vmesnika je implementiran z eno aktivnostjo.

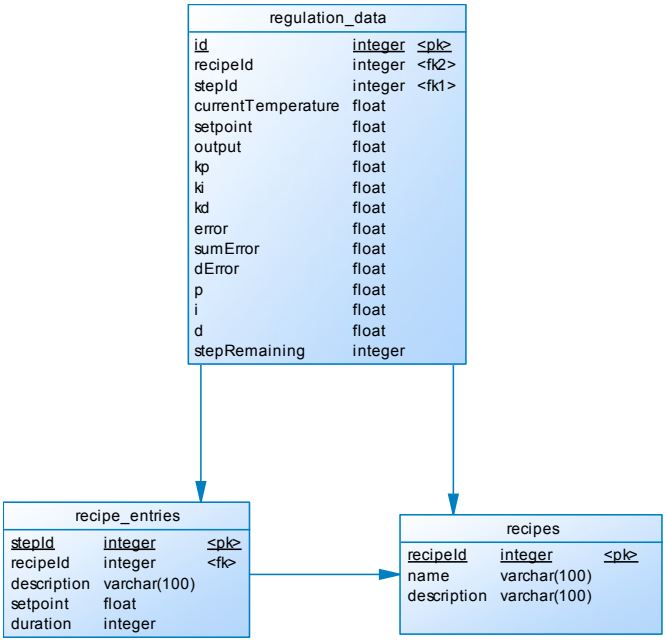
Glavna stran uporabniškega vmesnika je prikazana na sliki Slika 14.



Slika 14: Pozdravno okno uporabniškega vmesnika aplikacije OpenTheBrew.

4.6.1 Struktura podatkovne baze

Za shranjevanje regulacijskih receptov in podatkov o regulaciji smo ustvarili tri tabele SQLite, ki so med seboj povezane, kot je razvidno s slike Slika 15.



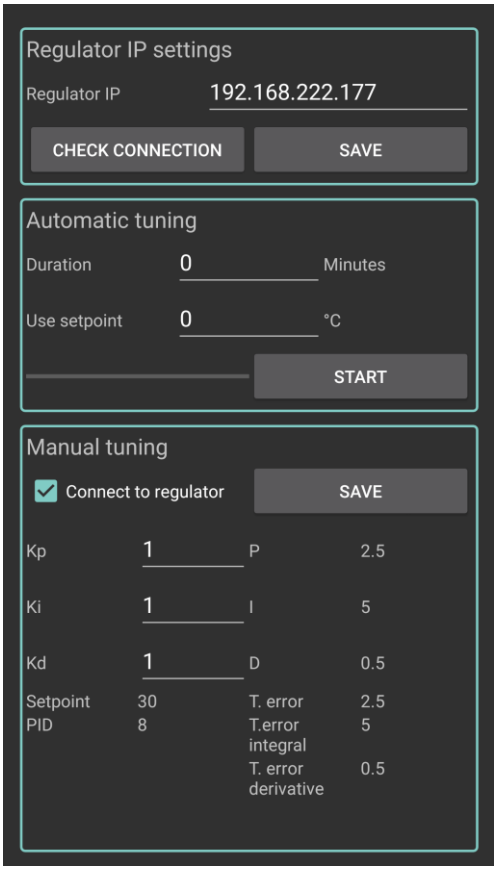
Slika 15: Shema uporabljene podatkovne baze v uporabniškem vmesniku Android.

Za hranjenje regulacijskih receptov smo uporabili tabeli recipes ter recipe_entries. Vsak vnos v tabeli recipes ima lahko enega ali več vnosov v tabeli recipe_entries. Zveznost med tema dvema tabelama ustvarimo preko tujega ključa recipeId v tabeli recipe_entries, ki je identičen primarnemu ključu vnosa v tabelo recipes. Na ta način enostavno povežemo posamezne korake v receptu z imenom in opisom recepta. Podatke iz te tabele smo uporabili za prikaz že ustvarjenih receptov v aplikaciji.

Tabelo regulation_data pa uporabljamo za hranjenje pridobljenih podatkov o regulaciji, ki jih vnesemo v tabelo po izvršitvi zahtevkov HTTP GET na strežnik HTTP platforme Arduino. Te podatke uporabimo v menujih regulacije in nastavitev za prikaz podatkov regulatorja preko grafikona ter tekstovnih polj.

4.6.2 Menu nastavitev

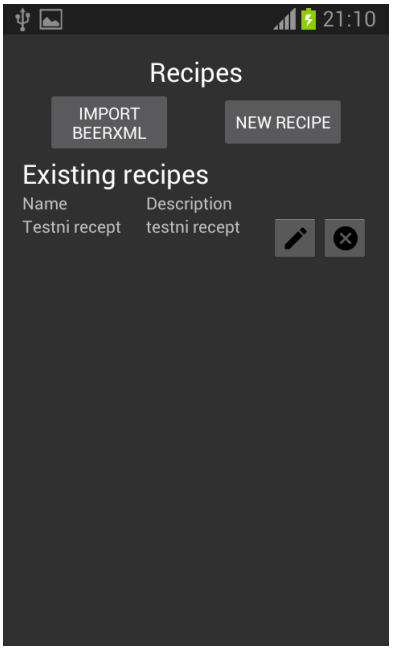
V tem menuju lahko ročno nastavljamo parametre regulatorja PID, spremljamo nastavitvene podatke o regulaciji, nastavimo naslov IP temperaturnega regulatorja ter zaženemo samodejno nastavitev parametrov PID preko metode Ziegler – Nichols z nihajnim preizkusom. Slika Slika 16 prikazuje izgled tega menuja.



Slika 16: Menu nastavitev uporabniškega vmesnika.

4.6.3 Menu receptov

V menuju receptov omogočamo uporabniku ustvarjanje, urejanje ter brisanje regulacijskih receptov, kot je prikazano na sliki Slika 17.



Slika 17: Slika menuja receptov.

Vsak regulacijski recept je sestavljen iz opisa recepta in vsaj enega regulacijskega koraka. Vsakemu koraku lahko določimo želeno temperaturo in trajanje ter opis posameznega koraka.

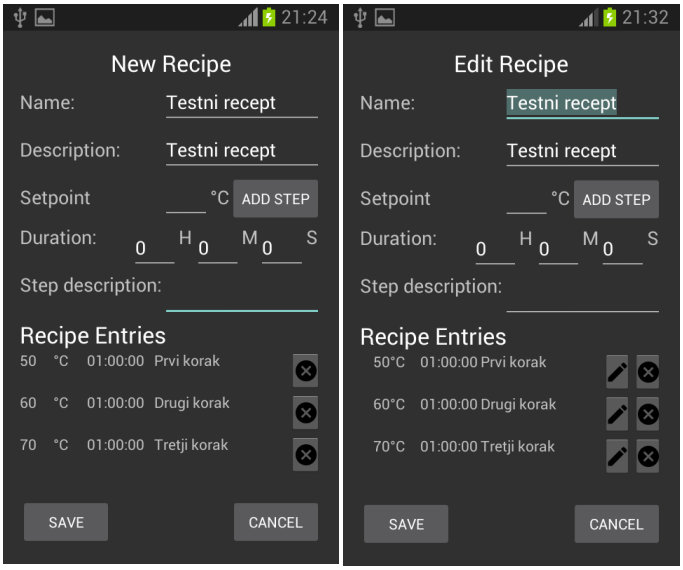
Število ustvarjenih receptov ni omejeno. Vsak recept pa lahko vsebuje največ pet regulacijskih korakov, saj smo omejeni s količino prostega pomnilnika na platformi Arduino.

Dodatno lahko uvozimo že ustvarjene recepte, v formatu BeerXML, preko pritiska na gumb »IMPORT BEERXML«.

Na tem menuju lahko izbrišemo ali urejamo celotni recept preko gumbov na desni strani posameznega vnosa seznama receptov.

4.6.3.1 Nov recept in urejanje obstoječih receptov

Uporabnik lahko ustvari nov recept po pritisku na gumb »NEW RECIPE«, kar prikaže menu ustvarjanja novih receptov, prikazan na sliki Slika 18.



Slika 18: Prikaz menuja za ustvarjanje ali urejanje regulacijskih receptov.

Na tem menuju lahko dodamo opis celotnega recepta ter največ pet regulacijskih korakov. Za vsak korak najprej določimo želeno temperaturo in trajanje ter poljubno dodamo še opis. Po pritisku na gumb »ADD STEP« se vnešeni podatki dodajo kot nov korak v receptu. Kot smo že omenili smo tu omejeni na največ pet vnosov korakov za posamezen recept. Če želimo urejati ali izbrisati posamezen korak lahko to storimo preko gumbov na desni strani seznama korakov v receptu.

S pritiskom na gumb »SAVE« shranimo nov recept, ki bo na voljo za uporabo v naši rešitvi. Če nismo zadovoljni z našim receptom ali spremembami lahko preko pritiska na gumb »CANCEL« zvržemo vse spremembe na izbranem ali novem receptu.

4.6.3.2 Uvoz datoteke BeerXML

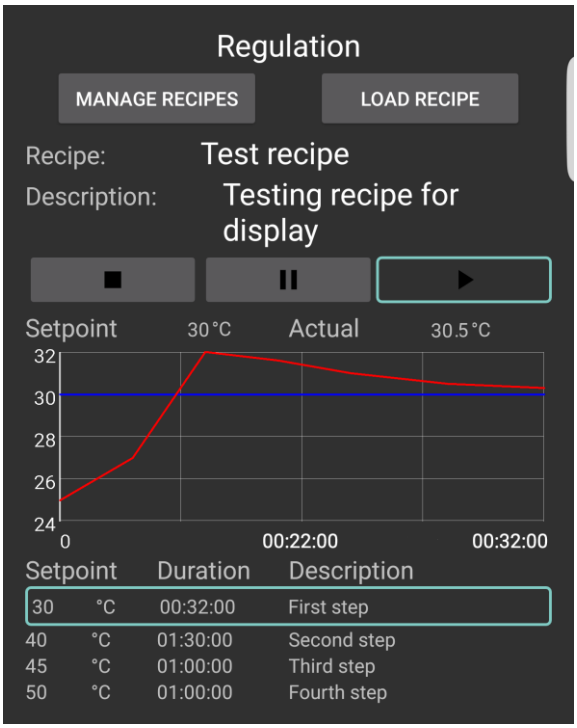
Uporabnik lahko po pritisku na gumb »IMPORT BEERXML« uvozi že narejen recept za pivo v formatu BeerXML s pomočjo knjižnice Material File Picker, opisane v poglavju o uporabljeni programski opremi. BeerXML je standardiziran format zapisa za izmenjavo receptov med pivovarji, ki vključuje razne informacije, potrebne za varjenje piva, ki v veliki meri presegajo potrebe temperaturne regulacije.

Vnosi v BeerXML so ločeni z oznakami XML, katere lahko enostavno uvozimo s pomočjo razčlenjevalnikov SAX. V naši implementaciji smo se iz obilice podatkov omejili le na uvoz podatkov, potrebnih za postopek drozganja, saj je ta regulacijsko najbolj zahteven.

BeerXML postopek drozganja razčleni na oznake z imenom MASH_STEPS, ki vsebujejo podrejene oznake STEP_TEMP in STEP_TIME. Dodatno vsebuje še razne druge oznake za količino vode, ki jo moramo v posameznem koraku dodati, nivo PH v drozgi itd.

4.6.4 Menu regulacija

Ta menu predstavlja jedro uporabniškega vmesnika. Na sliki Slika 19 spremljamo trenutno in preteklo stanje regulatorja, nastavljamo regulacijske recepte ter nadzorujemo potek izvajanja izbranega recepta.



Slika 19: Prikaz menuja regulacija.

Prikazan grafikon služi tudi za beleženje preteklih podatkov, saj nam v primeru, da povlečemo ekran v desno, prikaže pretekle podatke o regulaciji. Na grafikonu lahko opazimo dve črti. Modra nam označuje želeno temperaturo, medtem ko nam rdeča prikazuje dejansko temperaturo vode.

Pod grafikonom lahko vidimo seznam korakov izbranega regulacijskega recepta, kjer je trenutno aktiven korak obrobljen. Poleg želene temperature v stolpcu »Setpoint« lahko v stolpcu »Duration« vidimo koliko časa bo trenutni korak še aktiven ter v stolpcu »Description« opis posameznega koraka.

Na vrhu zaslona se nam izpišeta ime in opis izbranega recepta. Preko gumba »LOAD RECIPE« lahko izberemo enega izmed obstoječih receptov, ki ga želimo uporabiti za regulacijo. Preko gumba »MANAGE RECIPES« pa lahko enostavno dostopamo do menuja za ustvarjanje in urejanje receptov.

Ostanejo nam še trije gumbi neposredno nad grafikonom, ki služijo za zagon regulacije (gumb »PLAY«), trenutni ustavitvi odštevanja aktivnega koraka pri čemur se regulacija procesa na želeno temperaturo še vedno izvaja (gumb »PAUSE«), ter ustavljanje izvajanja recepta in ponastavitvi trenutnega koraka na začetnega (gumb »STOP«).

Poglavje 5 Sklepne ugotovitve

Pri razvoju rešitve nam je uspelo izpolniti začetni seznam zahtev za implementacijo. Ustvarili smo poceni in enostaven odprtokodni regulacijski sistem, ki je sposoben regulacije znotraj ene stopinje od želene temperature v stabilnem stanju. Seveda je ob stopničastih spremembah želene temperature potrebno nekaj časa, da se temperatura ustali okoli želene vrednosti, vendar ta potem s pomočjo regulacije PID dobro sledi želeni temperaturi. Sistem omogoča tudi uvoz standardnih receptov za drozganje v formatu BeerXML ter shranjevanje in izvajanje urnikov v skupni dolžini krepko dlje kot zahtevano. Zaradi te lastnosti ga lahko uporabimo tudi v drugih fazah varjenja, vendar pa moramo zaradi prostorskih omejitev te nastaviti v ločene regulacijske recepte.

Seveda pa naša rešitev v obstoječi obliki ni popolna. Možnosti za prihodnje izboljšave je še obilo, saj smo bili precej omejeni s skromno količino prostega pomnilnika na platformi Arduino Uno, tako da bi lahko z izbiro druge izvedbe Arduina implementirali več funkcionalnosti. Namesto uporabe le dveh zahtevkov HTTP GET bi lahko komunikacijski del razširili na več podprtih zahtevkov HTTP v obliki pravega vmesnika API. Tudi samo razčlenjevanje zahtevkov bi lahko v prihodnosti, z več časa, namenjenega optimizaciji, implementirali brez programske knjižnice String.h in nizov, temveč bi uporabili polja znakov, saj bi bil tak pristop boljši s stališča upravljanja pomnilnika. Razširili bi lahko tudi sam koncept regulacije na cel proces varjenja piva z avtomatizacijo pretoka surovin skozi stopnje procesa, tako da bi zmanjšali potrebno ročno delo uporabnikov.

Poleg izboljšav na platformi Arduino bi lahko izboljšali tudi prikaz uporabniškega vmesnika, saj je obstoječi minimalističen, vendar funkcionalen. Dodali bi lahko tudi razširjeni mehanizem za upravljanje receptov, tako da bi lahko polno izkoristili prednosti formata BeerXML. Naši načrti za prihodnost sicer niso omejeni le na to diplomsko delo, saj je bila izdelava sklada OpenTheBrew namenjena odprtokodni skupnosti, ker bo projekt javno dostopen na portalu GitHub po objavi tega diplomskega dela. S tem bomo lahko izkoristili predloge in prispevke zainteresirane odprtokodne javnosti v rešitvi, ki bo občutno cenejša od že obstoječih na trgu. Tako bomo dodali še en kamenček v rastoč mozaik ekosistema interneta stvari.

Seznam slik in tabel

Literatura

[1] Postopek priprave piva po metodi All Grain. *Pivovarnar.si* [Online]. Dosegljivo: http://pivovarnar.si/postopek-varjenja-piva/.

[2] Brewing. *Wikipedia.org* [Online]. Dosegljivo: https://en.wikipedia.org/wiki/Brewing.

[3] BrewPi 0.3.0 documentation. *BrewPi* [Online]. Dosegljivo: http://docs.brewpi.com/.

[4] BrewBit. *BrewBit* [Online]. Dosegljivo: http://brewbit.com/.

[5] Arduino Uno. *Arduino* [Online]. Dosegljivo: https://www.arduino.cc/en/Main/ArduinoBoardUno.

[6] Arduino Ethernet Shield. *Arduino* [Online]. *Dosegljivo:* https://www.arduino.cc/en/Main/ArduinoEthernetShield.

[7] Temperature Sensing using DS18B20 Digital Sensors. *OpenEnergyMonitor* [Online]. Dosegljivo: https://learn.openenergymonitor.org/electricity-monitoring/temperature/DS18B20-temperature-sensing.

[8] DS18B20 Datasheet. *Maxim Integrated* [Online]. Dosegljivo: https://datasheets.maximintegrated.com/en/ds/DS18B20.pdf.

[9] B. Zupančič. *Vodenje Sistemov.* Ljubljana: Založba FE in FRI, 2013.

[10] PID Theory Explained. *National Instruments* [Online]. Dosegljivo: http://www.ni.com/white-paper/3782/en/.

[11] S. Strmičnik, Celostni pristop k računalniškemu vodenju procesov. Ljubljana: Fakulteta za elektrotehniko, 1998.

[12] Finn Haugen, *PID Control*, Trondheim: Tapir Academic Press, 2004.

[13] Pulse-width modulation. *Wikipedia.org* [Online]. Dosegljivo: http://en.wikipedia.org/wiki/Pulse-width_modulation.